One-pass Context-based Tableaux Systems for CTL and ECTL

Alexander Bolotov
University of Westminster,
[W1W 6UW, London, UK.]
A.Bolotov@westminster.ac.uk

Paqui Lucio
University of the Basque Country
[P. Manuel de Lardizabal, 1, 20018-San Sebastián, Spain.]
paqui.lucio@ehu.eus

Montserrat Hermo
University of the Basque Country
[P. Manuel de Lardizabal, 1, 20018-San Sebastián, Spain.]
montserrat.hermo@ehu.es

Alex Abuin
Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA)
[P. J. M. Arizmendiarrieta, 2 20500 Arrasate-Mondragón Gipuzkoa, Spain]
aabuin@ikerlan.es

Abstract

When building tableau for temporal logic formulae, applying a two-pass construction, we first check the validity of the given tableaux input by creating a tableau graph, and then, in the second 'pass', we check if all the eventualities are satisfied. In one-pass tableaux checking the validity of the input does not require these auxiliary constructions. This paper continues the development of one-pass tableau method for temporal logics introducing tree-style one-pass tableau systems for Computation Tree Logic (CTL) and shows how this can be extended to capture Extended CTL (ECTL). A distinctive feature here is the utilisation, for the core tableau construction, of the concept of a context of an eventuality which forces its earliest fulfilment. Relevant algorithms for obtaining a systematic tableau for these branching-time logics are also defined. We prove the soundness and completeness of the method. With these developments of a tree-shaped one-pass tableau for CTL and ECTL, we have formalisms which are well suited for the automation and are amenable for the implementation, and for the formulation of dual sequent calculi. This brings us one step closer to the application of one-pass context-based tableaux in certified model checking for a variety of CTL-type branching-time logics.

2012 ACM Subject Classification Theory of computation –> Modal and temporal logics

Keywords and phrases Temporal logic, fairness, expressiveness, branching-time.

Digital Object Identifier 10.4230/LIPIcs...

Funding *Paqui Lucio*: This author has been supported by the European Union (FEDER funds) under grant TIN2017-86727-C2-2-R, and by the University of the Basque Country under Project LoRea GIU18-182.
Montserrat Hermo: This author has been supported by the European Union (FEDER funds) under grant TIN2017-86727-C2-2-R, and by the University of the Basque Country under Project LoRea GIU18-182.

1 Introduction.

In this paper we continue our investigation of tableaux-based deductive techniques for temporal logic having in mind its potential application in model checking, more specifically, in certified model checking [17]. There are two known ways to build tableau constructions for temporal logic formulae (for the survey of tableau method for temporal logic we refer an interested reader to [14]). So called two-pass constructions check the validity of the given tableaux input in two passes - once a tableau

graph has been obtained, in the second 'pass' we check if all the eventualities are satisfied or not. On the other hand, in one-pass tableaux checking the validity of the input does not require these auxiliary constructions. This paper continues the development of one-pass tableau method for temporal logics, this time specifically interested in Computation Tree Logic (CTL) originally introduced in [6] and Extended Computation Tree Logic (ECTL) introduced in [10]. The distinctive feature of our method is that the core tableau construction is based on the concept of *a context of an eventuality*, which is simply the set of formulae that 'accompanies' the eventuality in the label of the node. Our tableau rules that involve context force the earliest fulfilment of eventualities. In previous works such a context-based one-pass tableaux approach has been developed for propositional linear-time temporal logic, PLTL [12], and for the branching-time logic ECTL$^{\#}$ [4], which introduces a new class of fairness constraints utilising the 'until' temporal operator. It has also been shown how, in linear-time case, the method can be invoked as part of the certified model checking for PLTL, being "fortified" by a SAT solver. Aiming at similar developments for branching-time case, in particular for CTL, we make two observations.

Firstly, although the PLTL satisfaction of a property φ can be reduced to check if a free-model (i.e. the complete transition system) satisfies $\neg\varphi$ (since any counter-model of $\neg\varphi$ is a model of φ), let us note that the CTL satisfiability problem cannot be reduced to the CTL model checking on a free-model. In particular, a model checking algorithm for CTL properties (for example [5] (implemented in NuSMV)) cannot be adapted for testing CTL satisfiability. Indeed, the model checking problem for CTL is known to be P-complete [7], whereas the satisfiability problem for CTL is EXPTIME-complete [11]. On the contrary, any decision procedure of CTL satisfiability can be used to perform model checking tasks. Also, certified model checking (cf. [17]) which aims to generate proofs as certificates of true properties. as well as counterexamples for false properties would be a very useful development. Hence, some kind of proof system, e.g. sequent calculus, is required to check the proof certificates in branching-time case which will expand similar development for linear-time setting [2].

Secondly, one would assume that it is already covered by the technique developed for a richer logic - ECTL$^{\#}$. However, the application of such model checking procedure for CTL simply based on the existing one-pass tableaux for ECTL$^{\#}$ could become too "non-intuitive" due to the complexity of its rules. We also note that the distinguished (and unavoidable) feature of one-pass technique for ECTL$^{\#}$ is the utilisation of two types of context, unlike in case of PLTL. Here so called 'outer' (similar to PLTL) context is a collection of state formulae, and is complemented by so called inner context, a collection of path formulae. Subsequently, in this paper, we study the simplification of the method in [4] to a simpler branching temporal logic CTL. Note that in our tableau method for CTL, similarly to PLTL, we only need the "outer" context, yet, similar to ECTL$^{\#}$ the generated tableaux are AND-OR trees. Our results provide an intuitive tableau method that serves as a decision procedure of CTL satisfiability and can also be used in certified model checking of CTL properties. The method presented in the paper would enable a subsequent study and implementation of a certified model checking. With the development of tree-shaped one-pass tableaux for CTL and ECTL, this paper has proved the liveness of the approach which now covers both linear-time and a range of branching-time logics. Moreover, the results of this paper give us formalisms which are well suited for the automation and are amenable for the implementation, and for the formulation of a dual sequent calculi which brings us one step closer to the application of these developments in certified model checking.

Our extensive search for tableau methods for CTL has not shown a great variety of systems. For example, [3] presents a two-pass tableau, where in the first pass the tableau rules are applied creating a cyclic graph. In the second pass, the 'bad loops' are pruned (where a 'bad loop' is a loop containing some eventuality that is not fulfilled along it). This method was originally introduced in [9]. In [1, 13] the authors introduce a single-pass tableaux decision procedure for CTL. It is based on Schwendimann's one-pass procedure for PLTL. This tableau method uses an extra-logical

mechanism for collecting information of the set of formulae in the nodes, and passing it, to subsequent nodes along branches. The information about previously generated nodes is used to detect "bad loops" without constructing the whole graph. The complexity of this tableaux algorithm is double exponential deterministic time and needs exponential space. Finally, we note that we have not found an explicit formulation of a tableaux (one or two pass) method for ECTL.

To ensure that the presentation of quite technical details in the paper is clear and self-contained, we supply all major technical details in the text, otherwise, referring the reader to relevant sources for more details. This determines the following structure of the paper. In §2 we give CTL and ECTL syntax and semantics as sublogics of CTL*. The formulation of the tableau method is given in §3, where we first give some preliminaries and then overview the tableau construction as AND-OR tree and provide examples. A *systematic* tableau construction is introduced in §4. The soundness and completeness of our tableau method are proved in §6. Finally, in §7 we draw the conclusions and prospects of future work that the presented results open.

2 Syntax and Semantics of CTL and ECTL

The language of branching-time logic extends the language of classical propositional logic by future time temporal operators $\circ$ - 'at the next moment of time', $\Diamond$ - 'eventually', $\Box$ - 'always' and $\mathcal{U}$ - 'until', together with paths quantifiers A - 'for all paths' quantifier, and E - 'there exists a path' quantifier.

The hierarchy of CTL-type family of Branching-time logics (BTL) is defined by releasing restrictions on the concatenations of temporal operators and paths quantifiers resulting in distinguished for these logics classes of admissible state formulae. Thus, CTL itself requires every temporal operator to be preceded by a path quantifier, (and for example, cannot express fairness requiring at least the concatenation of $\Box$ and $\Diamond$). ECTL (Extended CTL) [9] enables simple fairness constraints but not their Boolean combinations. ECTL$^+$ [10] further extends the expressiveness of ECTL allowing Boolean combinations of temporal operators and ECTL fairness constraints (but not permitting their nesting). The logic ECTL$^\#$ [4] extends ECTL$^+$ by allowing the combinations $\Box(A\mathcal{U}B)$ or $A\mathcal{U}\Box B$, referred to as modalities $\Box\mathcal{U}$ and $\mathcal{U}\Box$. The logic CTL*, often considered as 'the full branching-time logic' overcomes all these restrictions on syntax. For the sake of generality, as all logics we are interested in are subsumed by CTL*, we first recall CTL* syntax and then, by restricting it, derive syntax for each of ECTL$^\#$, ECTL$^+$, ECTL and CTL.

▶ **Definition 1** (Syntax of CTL*). *Given* Prop *is a fixed set of propositions, and* $p \in$ Prop*, we define sets of state* (σ) *and path* (π) CTL* *formulae over* Prop *as follows:* $\sigma ::= \mathbf{T} \mid p \mid \sigma_1 \wedge \sigma_2 \mid \neg\sigma \mid \mathsf{E}\pi$ *and* $\pi_{\mathsf{CTL}^\star} ::= \sigma \mid \pi_1 \wedge \pi_2 \mid \neg\pi \mid \circ\pi \mid \pi\mathcal{U}\pi$. ■

A Kripke structure, $\mathcal{K}$, is a triple (S, R, L) where $S \neq \emptyset$ is a set of *states*, $R \subseteq S \times S$ is a total binary relation, called *the transition relation*, and $L : S \to 2^{\mathsf{Prop}}$ is a *labelling function*. Our Kripke structures are labelled directed graphs that correspond to Emerson's R-generable structures, i.e. the transition relation R is suffix, fusion and limit closed [8]. A *fullpath* x through a Kripke structure $\mathcal{K}$ is an infinite sequence of states $s_0, s_1, \ldots$ such that $(s_i, s_{i+1}) \in R$, for every $i \geq 0$. The notation $\mathcal{K} \restriction x(i)$ denotes a Kripke structure with the set of states of $\mathcal{K}$ restricted to those that are R-reachable from $x(i)$ and $\mathsf{fullpaths}(\mathcal{K})$ is the set of all fullpaths in $\mathcal{K}$. Given the structure $\mathcal{K} = (S, R, L)$, the relation $\models$, which evaluates path formulae in a given path x and state formulae at the state index i of the given path x is defined on atoms by $\mathcal{K}, x, i \models p$ iff $p \in L(x(i))$. Omitting standard definitions for Booleans, we present the relation $\models$ for temporal connectives and quantifier E:

$\mathcal{K}, x, i \models \mathsf{E}\pi$ iff there exists a path $y \in \mathsf{fullpaths}(\mathcal{K} \restriction x(i))$ such that $\mathcal{K}, y \models \pi$.
$\mathcal{K}, x \models \circ\pi$ iff $\mathcal{K}, x^{\geq 1} \models \pi$.
$\mathcal{K}, x \models \pi_1 \mathcal{U} \pi_2$ iff there exists $k \geq i$ with $\mathcal{K}, x^{\geq k} \models \pi_2$ and $\mathcal{K}, x^{\geq j} \models \pi_1$ for all $0 \leq j \leq k-1$.

where, given a fullpath $x = s_0, s_1, \ldots, s_k, \ldots \ (k \geq 0)$, we denote its finite prefix by the sequence $x^{\leq k} = s_0, s_1, \ldots, s_k$ and the suffix path $x^{\geq k} = s_k, s_{k+1}, \ldots$. For any set Σ of state formulae, $\mathcal{K}, x, i \models \Sigma$ iff $\mathcal{K}, x, i \models \sigma$, for all $\sigma \in \Sigma$. Moreover, if for any fullpath $x \in \mathsf{fullpaths}(\mathcal{K})$, we have $\mathcal{K}, x, 0 \models \Sigma$, then we simply write $\mathcal{K} \models \Sigma$.

For a state formula φ, the set of its models, $\mathsf{Mod}(\varphi)$, is formed by all triples $(\mathcal{K}, x, i)$ such that $\mathcal{K}, x, i \models \varphi$. Then φ is *satisfiable* ($\mathsf{Sat}(\varphi)$) if $\mathsf{Mod}(\varphi) \neq \emptyset$, otherwise φ is *unsatisfiable* ($\mathsf{UnSat}(\varphi)$) For state formulae φ and φ', if $\mathsf{Mod}(\varphi) = \mathsf{Mod}(\varphi')$ then φ and φ' are logically equivalent denoted as $\varphi \equiv \varphi'$. Satisfiability and logical equivalence are generalized to sets of state formulae Σ, in the natural way (formally by substituting φ with Σ in it).

For each of BTL logics ECTL$^{\#}$, ECTL^{+}, ECTL and CTL its syntax is defined over a fixed set of propositions Prop, such that the definition of state formulae is the same as for CTL* (Def. 1), and the eventuality $\Diamond A$ is the abbreviation for $\mathrm{T}\mathcal{U}A$. The specific for these logics restrictions on the CTL* grammar in Definition 1 generate the corresponding sets for path formulae, as in Def. 2.

▶ **Definition 2** (Paths formulae for ECTL$^{\#}$,ECTL^{+}, ECTL and CTL)**.**

$$\begin{array}{lcl}
\pi_{\mathsf{ECTL}^{\#}} & ::= & \sigma \mid \pi_1 \wedge \pi_2 \mid \neg\pi \mid \bigcirc\sigma \mid \sigma\mathcal{U}(\sigma \wedge \Diamond\sigma) \mid \Box(\sigma \vee \Box\sigma) \mid \sigma\mathcal{U}(\Box\sigma) \mid \Box(\sigma\mathcal{U}\sigma) \\
\pi_{\mathsf{ECTL}^{+}} & ::= & \sigma \mid \pi_1 \wedge \pi_2 \mid \neg\pi \mid \bigcirc\sigma \mid \sigma\mathcal{U}\sigma \mid \Box\sigma \mid \Box\Diamond\sigma \mid \Diamond\Box\sigma. \\
\pi_{\mathsf{ECTL}} & ::= & \sigma \mid \neg\pi \mid \bigcirc\sigma \mid \sigma\mathcal{U}\sigma \mid \Box\sigma \mid \Box\Diamond\sigma \mid \Diamond\Box\sigma. \\
\pi_{\mathsf{CTL}} & ::= & \sigma \mid \neg\pi \mid \bigcirc\sigma \mid \sigma\mathcal{U}\sigma \mid \Box\sigma.
\end{array}$$

■

Other usual operators can be derived from those introduced, in the standard way, in particular, the 'falsehood' constant $\mathrm{F} \equiv \neg\mathrm{T}$, the classical disjunction operator $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$ and the "release" operator $\varphi_1 \mathcal{R} \varphi_2 \equiv \neg(\neg\varphi_1 \mathcal{U} \neg\varphi_2)$, as well as the universal path quantifier $\mathsf{A}\varphi \equiv \neg\mathsf{E}\neg\varphi$. It is also known that $\Box\varphi \equiv \neg\Diamond\neg\varphi$, but, for technical convenience we define it as a primitive operator.

As we mentioned, we assume that the input to the tableaux is written in *Negation Normal Form*. Definitions 3, 4, and Proposition 5 introduce relevant framework for CTL-type branching-time logics.

▶ **Definition 3** (Literals)**.** *Let* Prop *be a fixed set of* CTL *(*ECTL*) propositions, and let* $\rho \in$ Prop*. Then the set of* CTL*(*ECTL*) literals is defined as* $Lit ::= \mathrm{F} \mid \mathrm{T} \mid \rho \mid \neg\rho$. ■

▶ **Definition 4** (Negation Normal form)**.** *For a given branching-time formula,* φ*, to obtain its Negation Normal Form abbreviated* $\mathsf{NNF}(\neg\varphi) \in \mathcal{F}_{\mathsf{Prop}}$*, push negation inwards until it only applies to literals based on the following equivalences (as left-to-right rewrite rules):*

$$\begin{array}{ll}
\neg\neg\varphi \equiv \varphi & \neg\bigcirc\varphi \equiv \bigcirc\neg\varphi \\
\neg\mathsf{E}\varphi \equiv \mathsf{A}\neg\varphi & \neg\mathsf{A}\varphi \equiv \mathsf{E}\neg\varphi \\
\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi & \neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi \\
\neg\Diamond\varphi \equiv \Box\neg\varphi & \neg\Box\varphi \equiv \Diamond\neg\varphi \\
\neg(\varphi\mathcal{U}\psi) \equiv (\neg\varphi)\,\mathcal{R}\,(\neg\psi) & \neg(\varphi\,\mathcal{R}\,\psi) \equiv (\neg\varphi)\mathcal{U}(\neg\psi)
\end{array}$$

■

It is trivial to see that φ and $\mathsf{nnf}(\varphi)$ are logically equivalent and that all logics, CTL, ECTL, ECTL^{+}, ECTL$^{\#}$ and CTL* are closed under negation.

▶ **Proposition 5** (CTL, ECTL, ECTL^{+}, ECTL$^{\#}$ and CTL* are closed under Negation)**.** *For any* $\varphi \in \mathcal{F}_{\mathsf{Prop}}$*, we also have* $\mathsf{NNF}(\neg\varphi) \in \mathcal{F}_{\mathsf{Prop}}$*. Moreover, the negation of a state (resp. path) formula is a state (resp. path) formula.* ■

For simplicity, we will write $\neg\varphi$ instead of $\mathsf{NNF}(\neg\varphi)$. Also, for a finite set $\Delta = \{\varphi_1, \ldots, \varphi_n\}$, we let $\mathsf{NNF}(\neg\bigwedge_{i=1}^{n}\varphi_i) = \neg\Delta$. In what follows, we assume that every formula is in nnf, in particular, by $\neg\sigma$, we abbreviate $\mathsf{NNF}(\neg\sigma)$. The symbol Q stands for any path quantifier: A or E.

While nesting of 'pure path formulae' is totally unrestricted in CTL*, it is now restricted in its sublogics by relevant grammar cases for paths formulae. For example, a CTL* formula

$$\mathsf{A}\Diamond(\bigcirc p \wedge \mathsf{E}\bigcirc\neg p) \tag{1}$$

is not an ECTL$^{\#}$ formula. Rewriting it as $\mathsf{A}(\mathrm{T}\mathcal{U}(\circ p \wedge \mathsf{E}\circ\neg p))$ we can see that $\circ p \wedge \mathsf{E}\circ\neg p$ is neither a state formula nor of the form $\Box\sigma$. Note also, that the validity of this indicative for CTL* formula is directly linked to the limit closure property [8]. Similarly, a ECTL$^{\#}$ formula $\mathsf{A}((p\mathcal{U}\Box q) \wedge (s\mathcal{U}\Box\neg q))$ is not an ECTL^{+} formula because $p\mathcal{U}\Box q$ and $s\mathcal{U}\Box\neg q$, hence their conjunction, are not admissible ECTL^{+} formulae. Further, a ECTL^{+} formula

$$\mathsf{E}(\Box\Diamond q \wedge \Diamond\Box\neg q) \tag{2}$$

(which is inconsistent) does not belong to ECTL syntax as $\Box\Diamond q \wedge \Diamond\Box\neg q$ is not an admissible ECTL path formula. Finally, the fairness constraint expressible in ECTL -

$$\mathsf{E}\Box\Diamond q \tag{3}$$

cannot be constructed in CTL syntax as every temporal operator in a CTL formula must be preceded by a path quantifier.

Note that it is important to distinguish the problem if a formula of a superlogic belongs to a sublogic and the problem if a formula of a superlogic can be expressed in a sublogic. For example, $\mathsf{E}(\Box\Diamond q \vee \Diamond\Box\neg q)$, similarly to formula (2) does not belong to ECTL but it is expressible in this logic, as $\mathsf{E}(\Box\Diamond q \vee \Diamond\Box\neg q) \equiv \mathsf{E}\Box\Diamond q \vee \mathsf{E}\Diamond\Box\neg q$ which is an ECTL formula if we define $\vee$ via $\wedge$.

BTL Logics	$\mathsf{E}\Box\Diamond q$	$\mathsf{E}(\Box\Diamond q \wedge \Diamond\Box\neg q)$	$\mathsf{A}((p\mathcal{U}\Box q) \vee (s\mathcal{U}\Box\neg r))$	$\mathsf{A}\Diamond(\circ p \wedge \mathsf{E}\circ\neg p)$	One-pass Tableaux
$\mathcal{B}(\mathcal{U},\circ)$ (CTL)	X	X	X	X	This paper
$\mathcal{B}(\mathcal{U},\circ,\Box\Diamond)$ (ECTL)	$\surd$	X	X	X	This paper
$\mathcal{B}^{+}(\mathcal{U},\circ,\Box\Diamond)$ (ECTL^{+})	$\surd$	$\surd$	X	X	$\surd$
$\boldsymbol{\mathcal{B}^{+}(\mathcal{U},\circ,\mathcal{U}\Box)}$ (ECTL$^{\#}$)	$\surd$	$\surd$	$\surd$	X	$\surd$
$\mathcal{B}^{\star}(\mathcal{U},\circ)$ (CTL*)	$\surd$	$\surd$	$\surd$	$\surd$	X

Figure 1 Classification of context-based tableaux systems for CTL-type logics and relevant difficult cases of concatenations of temporal operators and path quantifiers

Figure 1 represents BTL logics classified by their expressiveness using '$\mathcal{B}$' for 'Branching', followed by the set of only allowed modalities as parameters; $\mathcal{B}^{+}$ indicates admissible Boolean combinations of the modalities and $\mathcal{B}^{\star}$ reflects 'no restrictions' in either concatenations of the modalities or Boolean combinations between them following the notation initially proposed in [8] and further tuned in [16]. The top row of the figure represents the indicative formulae (1)-(3) for the listed logics. In the last column we use a short CTL* formula $\mathsf{A}\Diamond(\circ p \wedge \mathsf{E}\circ\neg p)$, not expressible by weaker logics. The last column in Figure 1 reflects the development of the context-based one-pass tableaux technique for CTL-type logics: the method has been developed for ECTL$^{\#}$ ([4] where the motivation was to tackle complex cases of fairness). In this paper we introduce the technique for CTL and ECTL, while the case of ECTL^{+} can be tackled effectively by the technique developed for ECTL$^{\#}$. Indeed, ECTL^{+} and ECTL$^{\#}$ have similar cases of the Boolean combination of eventualities in the scope of A and E, namely disjunctions of the eventualities in the scope of the A quantifier and conjunctions of eventualities in the scope of the E quantifier, see [4] for details. Thus, Figure 1 also reflects difficult, for context-based one-pass tableaux, syntactical cases of concatenations of temporal operators and path quantifiers. To tackle these cases, in addition to $\alpha - \beta$ rules, that are standard to the tableaux, novel β^{+}-rules which use the context to force the eventualities to be fulfilled as soon as possible, where introduced. As ECTL$^{\#}$ is more expressive than ECTL^{+} in allowing new type of fairness constraints that use the $\mathcal{U}$ operator, the relevant rules introduced in [4] would cover all difficult concatenations of operators in ECTL^{+}. Hence, simply treating the case of one-pass context-based tableaux for ECTL^{+} as solved by the relevant development for this reacher logic, we concentrate in this paper on bridging the gap in our roadmap in supplying BTL logics by this

technique - the development of the method for CTL and ECTL. The ultimate target of this roadmap - one-pass context-based tableaux for CTL* remains extremely difficult and open problem.

3 Context-based One-pass Tableau Method for CTL

We precede the presentation of the method by the introduction of a number of important concepts. Firstly, in analysing CTL-type logics, the concept of *basic modality* is very useful. It reflects the restrictions on forming the basic admissible combinations of temporal operators in the scope of a path quantifier. Recall that formulae of CTL and ECTL logics are written in nnf.

A basic modality of a CTL or ECTL logic is of the form QT, where T is a temporal operator so the structure QT is generated according the grammar rules for these logics in Def. 2. We can identify all basic modalities in a given formula ϕ by finding its most embedded modality(es), say M_1, then look at the next basic modality in which M_1 is embedded, etc.

▶ **Definition 6** (ECTL$^{\#}$,ECTL^{+}, ECTL and CTL Basic Modalities)**.**

$$\begin{array}{lll} \mathsf{M_{ECTL}} & ::= & c \mid \mathsf{Q}\circ\mathsf{M} \mid \mathsf{Q}(\mathsf{M}\mathcal{U}\mathsf{M}) \mid \mathsf{Q}\Box\mathsf{M} \mid \mathsf{Q}\Box\Diamond\mathsf{M} \mid \mathsf{Q}\Diamond\Box\mathsf{M}. \\ \mathsf{M_{CTL}} & ::= & c \mid \mathsf{Q}\circ\mathsf{M} \mid \mathsf{Q}(\mathsf{M}\mathcal{U}\mathsf{M}) \mid \mathsf{Q}\Box\mathsf{M}. \end{array}$$

■

where c stands for a purely classical formula (we can consider a purely classical formula as a zero-degree basic modality) and M stands for any basic modality of CTL in the definition of $\mathsf{M_{CTL}}$ and of ECTL in the definition of $\mathsf{M_{ECTL}}$. For example, for CTL, its basic modalities would be structures $\mathsf{Q}\circ$, $\mathsf{Q}\mathcal{U}$, and $\mathsf{Q}\Box$ while for ECTL these will be $\mathsf{Q}\circ$, $\mathsf{Q}\mathcal{U}$, $\mathsf{Q}\Box$, $\mathsf{Q}\Diamond\Box$ and $\mathsf{Q}\Box\Diamond$. If we analyse a CTL formula $\mathsf{E}\circ\mathsf{A}\circ p$ then the most embedded basic modality would be $\mathsf{A}\circ p$, which is embedded in $\mathsf{E}\circ M_1$. In what follows, every CTL modality $\mathsf{Q}\mathcal{U}$ or $\mathsf{Q}\Diamond$ and ECTL modality $\mathsf{Q}\Diamond\Box$ is called *eventuality*.

The tableau rules presented in the next section, for $\mathsf{Q}\Diamond$, $\mathsf{Q}\Box$, $\mathsf{Q}\mathcal{U}$ and $\mathsf{Q}\mathcal{R}$ are based on fixpoint characterisation of basic CTL modalities. For example, $\mathsf{E}\Box$, $\mathsf{A}\Box$, $\mathsf{E}(\phi\,\mathcal{R}\,\psi)$ and $\mathsf{A}(\phi\,\mathcal{R}\,\psi)$ can be represented as maximal fixpoints (equations (4)) while $\mathsf{E}\Diamond$, $\mathsf{A}\Diamond$, $\mathsf{E}(\phi\mathcal{U}\psi)$ and $\mathsf{A}(\phi\mathcal{U}\psi)$ as minimal fixpoints (equations (5)). In the equations below ν and μ stand for 'minimal fixpoint' and 'maximal fixpoint' operators, respectively.

$$\begin{array}{ll} \mathsf{E}\Box\phi = \nu\rho(\phi \wedge \mathsf{E}\circ\rho) & \mathsf{E}(\phi\,\mathcal{R}\,\psi) = \nu\rho(\psi \wedge (\phi \vee \mathsf{E}\circ\rho)) \\ \mathsf{A}\Box\phi = \nu\rho(\phi \wedge \mathsf{A}\circ\rho) & \mathsf{A}(\phi\,\mathcal{R}\,\psi) = \nu\rho(\psi \wedge (\phi \vee \mathsf{A}\circ\rho)) \end{array} \tag{4}$$

$$\begin{array}{ll} \mathsf{E}\Diamond\phi = \mu\rho(\phi \vee \mathsf{E}\circ\rho) & \mathsf{E}(\phi\mathcal{U}\psi) = \mu\rho(\psi \vee (\phi \wedge \mathsf{E}\circ\rho)) \\ \mathsf{A}\Diamond\phi = \mu\rho(\phi \vee \mathsf{A}\circ\rho) & \mathsf{A}(\phi\mathcal{U}\psi) = \mu\rho(\psi \vee (\phi \wedge \mathsf{E}\circ\rho)) \end{array} \tag{5}$$

Thus, for example, $\mathsf{E}\Box\phi$ is the maximal solution for $\rho = (\phi \wedge \mathsf{E}\circ\rho)$ while $\mathsf{E}(\phi\mathcal{U}\psi)$ is the minimal solution for $\rho = (\psi \vee (\phi \wedge \mathsf{E}\circ\rho))$. The fixpoint characterisation of basic CTL and ECTL modalities as maximal or minimal fixpoints give rise their analytical classification as 'alpha' or 'beta' formulae which are associated, in the tableau construction with 'alpha' and 'beta' rules. Thus, $\mathsf{Q}\Box$, and $\mathsf{Q}\mathcal{R}$ as maximal fixpoints are classified as 'alpha' formulae and $\mathsf{Q}\Diamond$ and $\mathsf{Q}\mathcal{U}$ as 'beta' formulae, which is also reflected in the known equivalences:

$$\begin{array}{ll} \mathsf{E}\Box\phi = \phi \wedge \mathsf{E}\circ\mathsf{E}\Box\phi & \mathsf{E}(\phi\,\mathcal{R}\,\psi) = \psi \wedge (\phi \vee \mathsf{E}\circ\mathsf{E}(\phi\,\mathcal{R}\,\psi)) \\ \mathsf{A}\Box\phi = \phi \wedge \mathsf{A}\circ A\Box\phi & \mathsf{A}(\phi\,\mathcal{R}\,\psi) = \psi \wedge (\phi \vee \mathsf{A}\circ\mathsf{A}(\phi\,\mathcal{R}\,\psi)) \end{array} \tag{6}$$

$$\begin{array}{ll} \mathsf{E}\Diamond\phi = \phi \vee \mathsf{E}\circ\mathsf{E}\Diamond\phi & \mathsf{E}(\phi\mathcal{U}\psi) = \psi \vee (\phi \wedge \mathsf{E}\circ\mathsf{E}(\phi\mathcal{U}\psi)) \\ \mathsf{A}\Diamond\phi = \phi \vee \mathsf{A}\circ\mathsf{A}\Diamond\phi & \mathsf{A}(\phi\mathcal{U}\psi) = \psi \vee (\phi \wedge \mathsf{E}\circ\mathsf{E}(\phi\mathcal{U}\psi)) \end{array} \tag{7}$$

The tableau method aims to determine whether a given set of CTL state formulae is satisfiable or not. We precede the formal introduction of the technique by its informal overview. The initial node of the tableaux is labelled by a CTL formula in NNF. To expand the root, and any subsequent node, we apply one of the four types of rules: α- and β-rules, the 'next-state' rule, which reflects a 'jump' from a 'state' to a 'pre-state', and, finally, characteristic to our approach, β^+-rules, where the use

$$(\wedge)\ \frac{\Sigma, \sigma_1 \wedge \sigma_2}{\Sigma, \sigma_1, \sigma_2} \qquad (\mathsf{Q}\Box)\ \frac{\Sigma, \mathsf{Q}\Box\sigma}{\Sigma, \sigma, \mathsf{Q}\circ\mathsf{Q}\Box\sigma}$$

$$(\vee)\ \frac{\Sigma, \sigma_1 \vee \sigma_2}{\Sigma, \sigma_1 \mid \Sigma, \sigma_2} \qquad (\mathsf{Q}\mathcal{U})\ \frac{\Sigma, \mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2)}{\Sigma, \sigma_2 \mid \Sigma, \sigma_1, \mathsf{Q}\circ\mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2)}$$

$$(\mathsf{Q}\mathcal{R})\ \frac{\Sigma, \mathsf{Q}(\sigma_1 \mathcal{R} \sigma_2)}{\Sigma, \sigma_1, \sigma_2 \mid \Sigma, \sigma_2, \mathsf{Q}\circ\mathsf{Q}(\sigma_1 \mathcal{R} \sigma_2)} \qquad (\mathsf{Q}\Diamond)\ \frac{\Sigma, \mathsf{Q}\Diamond\sigma}{\Sigma, \sigma \mid \Sigma, \mathsf{Q}\circ\mathsf{Q}\Diamond\sigma}$$

Figure 2 ALPHA AND BETA RULES.

$$(\mathsf{Q}\circ)\ \frac{\Sigma, \mathsf{A}\circ\sigma_1, \ldots, \mathsf{A}\circ\sigma_\ell, \mathsf{E}\circ\sigma'_1, \ldots, \mathsf{E}\circ\sigma'_k,}{\sigma_1, \ldots, \sigma_\ell, \sigma'_1\ \&\ \ldots\ \&\ \sigma_1, \ldots, \sigma_\ell, \sigma'_k}\ \text{where}\ \Sigma\ \text{is a set of literals.}$$

Figure 3 NEXT-STATE RULE.

of the context (of an eventuality) is essential. The context, which is a collection of state formula accompanying the eventuality in the label of the node, forces eventualities to be fulfilled as soon as possible. We apply the α, β, β^+- rules repeatedly until we reach a node which is labelled by $\mathbf{F}$ or by an inconsistent set of formulae, or a node whose labels have already occurred within the path under consideration. In the former case the expansion of the given branch terminates with $\bot$ as its leaf. In the latter case, a repetitive node in the branch means that the input formula is satisfied forever, and we select another eventuality (if any).

▶ **Definition 7** (Syntactically Consistent Set of Formulae). *A set Σ of state formulae σ is syntactically consistent abbreviated as $\Sigma_\top$ if $\mathbf{F} \notin \Sigma$ and $\{\sigma, \neg\sigma\} \not\subseteq \Sigma$ for any σ; otherwise, Σ is inconsistent denoted as $\Sigma_\bot$.* ■

▶ **Definition 8** (Tableau, Consistent Node, Closed branch). *A tableau for a set of* CTL *state formulae Σ is a labelled tree T, where nodes are τ-labeled with sets of state formulae, such that the following two conditions hold: (i) The root is labelled by the set Σ. (ii) Any other node m is labelled with sets of state formulae as the result of the application of one of the rules in Figures 2, 3 and 4 to its parent node n. Given the applied rule is R, we term m an R-successor of n.*

A node n of T is consistent, abbreviated as $n_\top$, if its label, $\tau(n)$, is a syntactically consistent set of formulae (see Def. 7), else n is inconsistent, abbreviated as $n_\bot$. If a branch b of T, contains $n_\bot \in b$, then b is closed else b is open. ■

The rules presented in Figure 2 follow the standard for the tableaux classification of rules into α-rules and β-rules which is based on the analytic classification of formulae in the underlying logic. Equations (6)-(7) as we mentioned in the last section, reflect this analytic classification for CTL. Thus, if a node, n, in the tableau graph is labelled by a set of formulae, Σ, ϕ, and a designated formula for the application of tableau rules, ϕ, is an α-formula - $\mathsf{Q}\Box$ or $\mathsf{Q}\mathcal{R}$, then a corresponding $alpha$-rule applies, while we treat Σ as a (possibly empty) context for ϕ. If a node, n, in the tableau graph is labelled by a set of formulae Σ, ϕ, and ϕ is a β-formula - $\mathsf{Q}\Diamond$ or $\mathsf{Q}\mathcal{U}$ then a corresponding β-rule applies. while, again, we treat Σ as a context for ϕ. These applications of α-β rules generate a set of formulae in the conclusion as a label for the successor node, $n+1$, in case of an α-rule, or as labels of two successors of n, in case of a β-rule. In our construction of the tableaux for CTL we also utilise the derived rules, which are used for simplicity and to ease the understanding of the technique: $(\mathsf{Q}\Box)$ and $(\mathsf{Q}\Diamond)$. These two rules can be derived by applying the $(\mathsf{Q}\mathcal{R})$ rule (respectively $(\mathsf{Q}\mathcal{U})$ rule) to a CTL formula where $\mathsf{Q}\Box\sigma$ is represented as $Q(\mathbf{F}\,\mathcal{R}\,\sigma)$ (respectively $\mathsf{Q}\Diamond\sigma$ as $\mathsf{Q}(\mathbf{T}\mathcal{U}\sigma)$). A specific situation arises when a node, n, contains in its label what we call an *elementary set of formulae* - this

$$(\mathsf{Q}\mathcal{U})^{+}\ \frac{\Sigma, \mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2)}{\Sigma, \sigma_2 \ \mid\ \Sigma, \sigma_1, \mathsf{Q}\circ\mathsf{Q}((\sigma_1 \wedge \neg\Sigma')\mathcal{U}\sigma_2)} \qquad (\mathsf{Q}\Diamond)^{+}\ \frac{\Sigma, \mathsf{Q}\Diamond\sigma}{\Sigma, \sigma \ \mid\ \Sigma, \mathsf{Q}\circ\mathsf{Q}(\neg\Sigma'\mathcal{U}\sigma)}$$

where $\Sigma' = \Sigma \setminus \{\mathsf{Q}\Box\sigma \mid \mathsf{Q}\Box\sigma \in \Sigma\} \cup \{(\mathsf{Q}\circ)^i\mathsf{Q}\Box\sigma \mid i \geq 1 \text{ and } (\mathsf{Q}\circ)^i\mathsf{Q}\Box\sigma \in \Sigma\}$

Figure 4 BETA-PLUS RULES

set is exclusively formed by literals and formulae of the form $Q\circ\sigma$. This structure (an analogous construction, in the terminology of [18] was called a 'state') enables us to construct successors of n corresponding to 'pre-states' [18]. The following proposition states that we are guaranteed to reach such a tree structure, where the last node of every branch, at this stage of the construction, is a state.

▶ **Proposition 9.** *Any set of* CTL *state formulae has a tableau T such that the last node of every branch is labelled by an elementary set of state formulae.*

Proof (by tableau construction). Repeatedly apply to every expandable node any applicable α-rule or β-rule until all expandable node are elementary. Then, the next-state rule must be applied to every expandable node. ■

Proposition 9 enables the application of the so-called 'next-state rule' depicted in Figure 3. Applying this rule we split the current branch at node n where the set $\Sigma, \mathsf{A}\circ\sigma_1, \ldots, \mathsf{A}\circ\sigma_\ell, \mathsf{E}\circ\sigma'_1, \ldots, \mathsf{E}\circ\sigma'_k$ is satisfied, into k branches (i.e. into the number of branches equal to the number of $\mathsf{E}\circ$ constraints) where the successors of n along these branches are AND-successors, and are labelled each by a different set $\sigma_1, \ldots, \sigma_\ell, \sigma'_i$, for $i \in \{1, \ldots, k\}$. This rule splits branches in a 'conjunctive' way, and we use the symbol $\&$ to represent the generation of AND-successors of node n. Thus, the graphs generated by the tableaux with the application of the 'next-state' rule are AND-OR trees. When $\ell = k = 0$, the rule yields a unique new node labelled by the empty set. We assume that whenever $k = 0$ and $\ell > 0$, there exists a unique descendant labelled by $\sigma_1, \ldots, \sigma_\ell$.

▶ **Example 10.** If node n is labelled by the set $\mathsf{A}\circ p, \mathsf{A}\circ q, \mathsf{E}\circ\neg p, \mathsf{E}\circ r$, the application of the next-state rule would split the construction into two branches emanating from n, with two AND-successors of n labelled by the sets $p, q, \neg p$ and p, q, r, respectively. The former set is syntactically inconsistent following Definition 7. ■

The subsequent construction of a tableau, additionally, involves rules that are applied to so called 'uniform sets of formulae'.

▶ **Definition 11** (Uniform Set of Formulae). *A set of* CTL *state formulae Σ is uniform iff Σ is exclusively formed by literals and basic* CTL *modalities.* ■

Proposition 9 (previous section) ensures that at some stage of the tableau construction we are able to apply the 'next-state' rule. Applying Proposition 9 to construct a tableau with all its expandable nodes labelled by elementary sets of formulae, then applying the rule $(\mathsf{Q}\circ)$ (to every expandable node) and, finally, repeatedly applying (to every expandable node) the rules $(\wedge)$, and $(\vee)$. we can prove Proposition 12 which states that we can also reach the stage where last nodes of tableaux branches are labelled by uniform sets of formulae.

▶ **Proposition 12.** *Any set of* CTL *state formulae Σ has a tableau T such that labels of all its expandable nodes are uniform sets of formulae.*

▶ **Definition 13** (Uniform Tableau). *For any set Σ of* CTL *state formulae, the tableau for Σ provided by Proposition 12 is denoted* Uniform_Tableau*(Σ).* ■

We extend our set of tableau rules with the new two rules named as β^+-rules (Figure 4). Note that the $(\mathsf{Q}\Diamond)^+$ rule can be derived from the application of the $(\mathsf{Q}\mathcal{U})^+$ to the CTL formula $\mathrm{T}\mathcal{U}\sigma$. These rules,

similarly to β-rules, also split a branch into two branches. These two β^+-rules are the only rules in our system that make use of the context - their application force the eventualities to be satisfied as soon as possible (from the point of the tableau construction where an eventuality is selected to be expanded with a β^+ rule). The context is given by the sets Σ containing state formulae.

▶ **Definition 14** (Next-Step Variant). *A state formula* $\mathsf{Q}(\neg\Sigma'\mathcal{U}\sigma)$ *obtained by the application of a* β^+ *rule to formula* $\mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)$ *or* $\mathsf{Q}\Diamond\sigma$ *is called the next-step variant of* $\mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)$. ■

4 Systematic Tableau Construction

In this section we define an algorithm, $\mathcal{A}^{sys}$, that constructs a *systematic tableau*. Let us observe that, due to the rule $(\mathsf{Q}\bigcirc)$, any open tableau should have a collection of open branches including all the $(\mathsf{Q}\bigcirc)$-successors of any node labelled by an elementary sets of formulae. These collections of branches are called *bunches*. Any open bunch of the systematic tableau, constructed by the algorithm $\mathcal{A}^{sys}$ introduced in this section, enables the construction of a model for the initial set of formulae. The

Algorithm 1 Systematic Tableau Construction

```
1:  procedure SYSTEMATIC_TABLEAU(Σ0)          ▷ where Σ0: set of CTL state formulae
2:      if Σ0 is not uniform then T := Uniform_Tableau(Σ0)
3:      while T has at least one expandable node do
4:                                  ▷ Invariant: Any expandable node of T is labelled by an uniform set
5:          Choose any node ℓ in T such that τ(ℓ) is expandable          ▷ τ(ℓ) is uniform
6:          if there is no eventuality in τ(ℓ) then T := T[ℓ ←Uniform_Tableau(τ(ℓ))]
7:          else
8:              Eventuality_Selection(τ(ℓ))
9:              Apply_β+-rule(τ(ℓ))
10:             Let ℓ1, ℓ2 the two children of ℓ
11:             for i = 1 .. 2 do
12:                 if ℓi is expandable and τ(ℓi) is not uniform then
13:                     T := T[ℓi ←Uniform_Tableau(τ(ℓi))]
```

algorithm $\mathcal{A}^{sys}$ constructs an *expanded* tableau (see Definition 25) for the given input. $\mathcal{A}^{sys}$ applied to the input Σ_0, denoted as $\mathcal{A}^{sys}(\Sigma_0)$, returns a systematic tableau $\mathcal{A}^{sys}_{\Sigma_0}$. Intuitively, 'expanded' means 'complete' in the sense that any possible rule has been already applied at every node. Though the best way to implement this algorithm is a depth-first construction, for clarity, we formulate it as a breadth-first construction of a collection of subtrees. The procedure Uniform_Tableau, in the above Algorithm 1, was introduced in Definition 13 along with the notion of a uniform set of state formulae. The notation $T_1[\ell \leftarrow T_2]$ stands for the tableau T_1 where the expandable ℓ is substituted by the tableau T_2. In particular, $T[\ell \leftarrow$Uniform_Tableau$(\Sigma)]$ is the tableau T where the expandable ℓ is substituted by the Uniform_Tableau(Σ). Procedure Eventuality_Selection chooses an eventuality to which the corresponding beta-plus rule ($(\mathsf{Q}\mathcal{U})^+$ or $(\mathsf{Q}\Diamond)^+$) can be applied. Procedure Apply_β^+-rule(Σ) applies the corresponding β^+ rule to the selected eventuality and kept selected the next-step variant (Definition 14) of such eventuality.

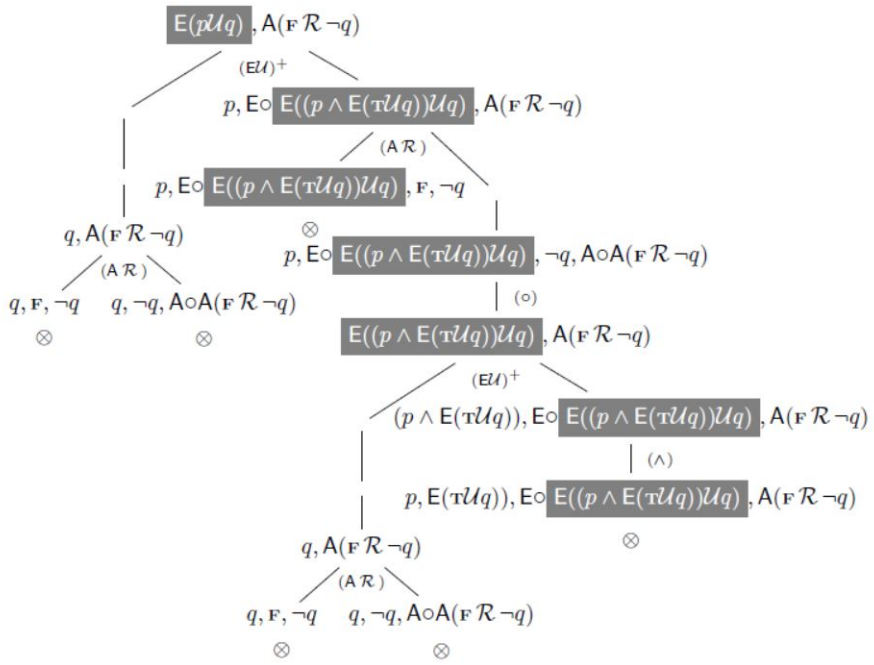

Figure 5 A closed tableau for $\{\mathsf{E}(p\mathcal{U}q), \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)\}$

▶ **Example 15.** In Figure 5 we depict the context-based tableau for the example exhibited in [1, 13]. The selected eventualities are in gray boxes. Note a direct correspondence between the tableaux - they have exactly the same nodes. The right-most branch, in our case, closes by (syntactical) inconsistency, likewise all the other branches. The difference is that, in this branch, the inconsistency comes from the use of the context in the selected eventuality. The corresponding branch in the tableau in [1, 13] is closed by the detection of a "bad loop". Intuitively, whenever the tableau in [1, 13] detects a "bad loop", our tableau is closed by contradiction. ■

The systematic tableau aims to obtain a loop-node that makes branches eventuality-covered.

▶ **Definition 16** (Loop-node). *Let b be a tableau branch and $n_i \in b$ $(0 \leq i)$. Then n_i is a loop-node if there exists $n_j \in b$ $(0 \leq j < i)$ such that $\tau(n_i) \subseteq \tau(n_j)$. We say that n_j is a companion node of n_i.* ■

▶ **Definition 17** (Eventuality-covered Branch). *A tableau branch $b = n_0, n_1, ..., n_i$ is eventuality-covered if n_i is a loop-node, with a companion node n_j $(0 \leq j < i)$, both labelled by a uniform set Σ such that every eventuality in $\tau(n_i)$ is selected in some node n_k $(j \leq k < i)$.* ■

The procedure Eventuality_Selection performs in some fair way that ensures that any open branch will ever be *eventuality-covered*.

▶ **Definition 18** (Non-expandable Node). *A node n is non-expandable if $\tau(n) = \Sigma_\perp$ or n is a loop-node of branch b which is eventuality-covered. Otherwise, n is expandable* ■

Consequently, an expandable node is either a node that is not a loop-node or a loop-node whose branch is not eventuality-covered.

▶ **Definition 19** (Bunch in a Tableau, Closed Bunch and Tableau). *A bunch b is a collection of branches that is maximal with respect to $(\mathsf{Q}\bigcirc)$-successor, i.e. every $(\mathsf{Q}\bigcirc)$-successor of any node in b is also in b. A bunch b is a closed bunch if, and only if, at least one of its branches is closed, otherwise it is open. A tableau is closed if, and only if, all its bunches are closed.* ■

Therefore, any open tableau has at least one open bunch, formed by one or more open branches. Open branches are ended in a loop node. Open bunches represent models, specifically *cyclic models* as defined in Appendix A. In Appendix B we provide an example of a tableau with AND-successor nodes, two open branches, but no open bunch.

5 Extending the Tableau from CTL to ECTL

In this section we explain a (relatively easy) way to extend the CTL tableau method to the more expressive logic ECTL. This is achieved by adding the new rules given in Figure 6.

$$(\mathsf{Q}\Box\Diamond)\ \frac{\Sigma, \mathsf{Q}\Box\Diamond\sigma}{\Sigma, \mathsf{Q}\Diamond\sigma, \mathsf{Q}\circ\mathsf{Q}\Box\Diamond\sigma} \qquad (\mathsf{Q}\Diamond\Box)\ \frac{\Sigma, \mathsf{Q}\Diamond\Box\sigma}{\Sigma, \mathsf{Q}\Box\sigma \mid \Sigma, \mathsf{Q}\circ\mathsf{Q}\Diamond\Box\sigma}$$

$$(\mathsf{Q}\Diamond\Box)^{+}\ \frac{\Sigma, \mathsf{Q}\Diamond\Box\sigma}{\Sigma, \mathsf{Q}\Box\sigma \mid \Sigma, \mathsf{Q}\circ\mathsf{Q}(\neg\Sigma'\mathcal{U}\mathsf{Q}\Box\sigma)} \quad \text{where } \Sigma' \text{ is as in Figure 4}$$

Figure 6 RULES FOR EXTENDING CTL TO ECTL.

The above rules correspond to the following logical equivalences for the basic modalities that extend CTL to ECTL: $\mathsf{Q}\Box\Diamond$ and $\mathsf{Q}\Diamond\Box$.

$$\mathsf{E}\Box\Diamond\sigma \equiv \mathsf{E}\Diamond\sigma \wedge \mathsf{E}\circ\mathsf{E}\Box\Diamond\sigma \qquad \mathsf{E}\Diamond\Box\sigma \equiv \mathsf{E}\Box\sigma \vee \mathsf{E}\circ\mathsf{E}\Diamond\Box\sigma$$
$$\mathsf{A}\Box\Diamond\sigma \equiv \mathsf{A}\Diamond\sigma \wedge \mathsf{A}\circ\mathsf{A}\Box\Diamond\sigma \qquad \mathsf{A}\Diamond\Box\sigma \equiv \mathsf{A}\Box\sigma \vee \mathsf{A}\circ\mathsf{A}\Box\Diamond\sigma$$

The rule $(\mathsf{Q}\Diamond\Box)^{+}$ is the context-based counterpart of the rule $(\mathsf{Q}\Diamond\Box)$. An example of ECTL tableau is given below.

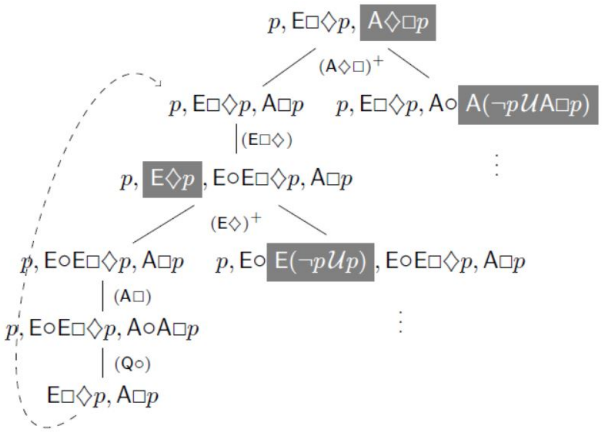

Figure 7 ECTL tableau for $\{p, \mathsf{E}\Box\Diamond p, \mathsf{A}\Diamond\Box p\}$ ($\vdots$ means this branch expansion is not depicted).

▶ **Example 20.** In Figure 7 we depict an open tableau that illustrates the application of the rules added to extend CTL to ECTL. We only show the left-most branch because it is an expanded open branch from which we can construct the model $\langle p \rangle^{\omega}$. ■

6 Soundness and Completeness

Here we sketch the correctness of our tableau method for CTL noting that similar results can be obtained for ECTL. Since CTL and ECTL are sublogics of ECTL$^{\#}$ and the tableau method is the adaptation of the method in [4], essentially, we adapt the correctness proof developed in [4].

▶ **Theorem 21** (Soundness). *Given any set of state formulae Σ, if there exists a closed tableau for Σ then Σ is unsatisfiable.*

Proof. (Sketch) After showing that tableau rules preserve satisfiability, we ensure that the root of the tableau is unsatisfiabile: any bunch in a closed tableau has at least one closed branch and the leaf of this closed branch is labelled by an inconsistent set. ∎

▶ **Theorem 22** (Refutational Completeness). *For any set of state formulae Σ_0, if* $\mathsf{UnSat}(\Sigma_0)$ *then there exists a closed tableau for Σ_0.*

To prove refutational completeness we firstly ensure that every open bunch in $\mathcal{A}^{sys}_{\Sigma_0}$ representents a model Σ_0 (Lemma 27).

▶ **Definition 23** (Stage). *Given a branch, b of a tableau T, a stage in T is every maximal subsequence of successive nodes $n_i, n_{i+1}, \ldots, n_j$ in b such that $\tau(n_k)$ is not a $(\mathsf{Q}\circ)$-child of $\tau(n_{k-1})$, for all k such that $i < k \leq j$.*

▶ **Definition 24** ($\alpha\beta^+$-saturated Stage). *A set of state formulae Ψ is $\alpha\beta^+$-saturated iff for all α-formula $\sigma \in \Psi$: Ψ contains all the formulae obtained by the application of an α-rule to σ, and for all β-formula $\sigma \in \Psi$, Ψ contains one of the two sets labelling the two children obtained by the application of either a β-rule or a β^+-rule to σ.* ∎

▶ **Definition 25** (Expanded Bunch and Tableau). *An open branch b is expanded if each stage $s \in \mathsf{stages}(b)$ is $\alpha\beta^+$-saturated and b is eventuality-covered. A bunch is expanded if all its open branches are expanded. A tableau is expanded if all its open bunches are expanded.* ∎

The following Proposition holds trivially by systematic tableau construction.

▶ **Proposition 26.** *Given any set of state formulae Σ_0, the systematic tableau $\mathcal{A}^{sys}_{\Sigma_0}$ is expanded.*

▶ **Lemma 27** (Model Existence). *For any expanded bunch H of $\mathcal{A}^{sys}_{\Sigma_0}$, there exists a Kripke structure $\mathcal{K}_H$ such that $\mathcal{K}_H \models \Sigma_0$.*

Proof. (Sketch) We define $\mathcal{K}_H = (S, R, L)$ such that $S = \bigcup_{b \in H} \mathsf{stages}(b)$ and for any $s \in S$: $L(s) = \{p \mid p \in \tau(n) \cap \mathsf{Prop} \text{ for some node } n \in s\}$; and R is the relation induced in $\mathsf{stages}(b)$ for each $b \in H$. Then, by structural induction in the modalities of CTL, we prove that any open branch of $\mathcal{A}^{sys}_{\Sigma_0}$ is a model of Σ_0. For that, Proposition 26 is crucial. ∎

Finally, we prove the refutational completeness of the tableau method.

Proof. (Sketch of the Proof of Theorem 22) Suppose the contrary, that there exists no closed tableau for Σ_0. Then the systematic tableau $\mathcal{A}^{sys}_{\Sigma_0}$ would be open and there would be at least one expanded bunch H in $\mathcal{A}^{sys}_{\Sigma_0}$. By Lemma 27, $\mathcal{K}_B \models \Sigma_0$. Consequently, Σ_0 would be satisfiable. ∎

7 Conclusion

We introduced a one-pass context-based tableau method for temporal logics CTL and ECTL, providing the soundness and completeness arguments and illustrating the method on a number of examples. The distinctive feature of the method presented in the paper, is that the core tableau construction is based on the concept of a context of an eventuality. The method developed in the paper is much simpler than the analogous technique obtained earlier for a richer logic - ECTL$^{\#}$ where two types of context (both outer and inner contexts) are used. Our construction only uses the "outer" context, however, similar to ECTL$^{\#}$, generates tableaux as AND-OR trees.

Our results provide intuitive tableau methods that serve as decision procedures of CTL and ECTL satisfiability. The results of this paper also give us formalisms which are well suited for the automation and are amenable for the implementation, and for the formulation of a dual sequent calculi. All these enable a potential application of the developed tableau methdods in certified model checking.

References

1 Pietro Abate, Rajeev Goré, and Florian Widmann. One-pass tableaux for computation tree logic. In Nachum Dershowitz and Andrei Voronkov, editors, Logic for Programming, Artificial Intelligence, and Reasoning, pages 32–46, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

2 Alex Abuin, Alexander Bolotov, Unai Díaz-de-Cerio, Montserrat Hermo, and Paqui Lucio. Towards certified model checking for PLTL using one-pass tableaux. In Johann Gamper, Sophie Pinchinat, and Guido Sciavicco, editors, 26th International Symposium on Temporal Representation and Reasoning, TIME 2019, October 16-19, 2019, Málaga, Spain, volume 147 of LIPIcs, pages 12:1–12:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.TIME.2019.12.

3 Mordechai Ben-Ari, Amir Pnueli, and Zohar Manna. The temporal logic of branching time. Acta Inf., 20(3):207–226, September 1983. doi:10.1007/BF01257083.

4 Alexander Bolotov, Montserrat Hermo, and Paqui Lucio. Branching-time logic ectl# and its tree-style one-pass tableau: Extending fairness expressibility of ectl+. Theoretical Computer Science, 813:428 – 451, 2020. doi:https://doi.org/10.1016/j.tcs.2020.02.015.

5 J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 1020 states and beyond. Information and Computation, 98(2):142 – 170, 1992. URL: http://www.sciencedirect.com/science/article/pii/089054019290017A, doi:https://doi.org/10.1016/0890-5401(92)90017-A.

6 E. M. Clarke and E. A. Emerson. Using Branching Time Temporal Logic to Synthesise Synchronisation Skeletons. Science of Computer Programming, pages 241–266, 1982.

7 Edmund M. Clarke, E. Allen Emerson, and Aravinda P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst., 8(2):244–263, 1986.

8 E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, Handbook of Theoretical Computer Science (Vol. B), pages 995–1072. MIT Press, Cambridge, USA, 1990.

9 E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. Journal of Computer and System Sciences, 30(1):1 – 24, 1985.

10 E. Allen Emerson and Joseph Y. Halpern. Sometimes and not never revisited: On branching versus linear time temporal logic. J. ACM, 33(1):151–178, 1986.

11 E.Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. Journal of Computer and System Sciences, 30(1):1 – 24, 1985. URL: http://www.sciencedirect.com/science/article/pii/0022000085900017, doi:https://doi.org/10.1016/0022-0000(85)90001-7.

12 Jose Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro, and Fernando Orejas. Dual systems of tableaux and sequents for PLTL. Journal of Logic and Algebraic Programming, 78(8):701–722, 2009.

13 Rajeev Goré. And-or tableaux for fixpoint logics with converse: Ltl, ctl, PDL and CPDL. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings, volume 8562 of Lecture Notes in Computer Science, pages 26–45. Springer, 2014. doi:10.1007/978-3-319-08587-6_3.

14 Rajeev Goré. Tableau methods for modal and temporal logics. Handbook of Tableau Methods, 01 1995. doi:10.1007/978-94-017-1754-0_6.

15 Ryo Kashima. An axiomatization of ECTL. J. Log. Comput., 24(1):117–133, 2014. doi:10.1093/logcom/ext005.

16 Nicolas Markey. Temporal logics. Course notes, Master Parisien de Recherche en Informatique, Paris, France, 2013. URL: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/NM-coursTL13.pdf.

17 Alain Mebsout and Cesare Tinelli. Proof certificates for smt-based model checkers for infinite-state systems. In Proceedings of the 16th Conference on Formal Methods in Computer-Aided Design, FMCAD '16, pages 117–124, 2016. doi:10.1109/FMCAD.2016.7886669.

18 Pierre Wolper. The tableau method for temporal logic: An overview. Logique Et Analyse, 28(110-111):119–136, 1985.

A Interpretation of CTL-type Logics Over Cyclic Structures

In this appendix we define cyclic models and discuss their ability to characterize satisfiability in branching temporal logics.

▶ **Definition 28** (Cyclic Sequence, Cyclic Path and Cyclic Kripke structure)**.** *Let z be a finite sequence of states $z = s_0, s_1, \ldots, s_j$ such that, for every $0 \leq k < j$, $(s_k, s_{k+1}) \in R$. Then, z is cyclic iff there exists s_i, $0 \leq i \leq j$ such that $(s_j, s_i) \in R$. Let z be a finite cyclic sequence, the subsequence $s_i, \ldots, s_j$ of z is called a loop and s_i is called the cycling element. We denote the loop as $\langle s_i, \ldots, s_j \rangle^\omega$. A cyclic path over z is an infinite sequence $\mathsf{path}(z) = s_0, s_1, \ldots, s_{i-1} \langle s_i, s_{i+1}, \ldots, s_j \rangle^\omega$. A Kripke structure $\mathcal{K}$ is cyclic if every fullpath is a cyclic path over a cyclic sequence of states.* ■

Cyclic paths are also known as *ultimately periodic* paths.

The fact that CTL (ECTL) satisfiability can be reduced to the interpretation over cyclic models only is derived from the existence of the finite model property [9], see also [15]. Hence, for any CTL (ECTL) formula φ, such that $\mathsf{Mod}(\varphi) \neq \emptyset$, there always exists a model $\mathcal{K} \in \mathsf{Mod}(\varphi)$ such that $\mathcal{K}$ is cyclic. Therefore, when speaking about the satisfiability in CTL (hence CTL) we can consider *cyclic* Kripke structures.

B An Example with Bunches and Loop Nodes

▶ **Example 29.** Figure 8 presents a closed tableau for $\{p, \mathsf{A}\Box\mathsf{E}\circ p, \mathsf{A}\Box\mathsf{E}\circ\neg p, \mathsf{A}\Diamond\neg p\}$. We use large circles to represent the generation of bunches. Every bunch in Figure 8 is closed, but at the same time contains an open branch. There are two open branches that represent models of $p, \mathsf{A}\Box\mathsf{E}\circ\neg p, \mathsf{A}\Diamond\neg p$, but the other branches (in the two bunches) reveal that $\mathsf{A}\Box\mathsf{E}\circ p$ can not be satisfied (due to the remaining formulae). Indeed, the “bad loop detection approach" would create a "bad loop" branch similar to the largest branch of our tableau in which $\circ p$ is satisfied.

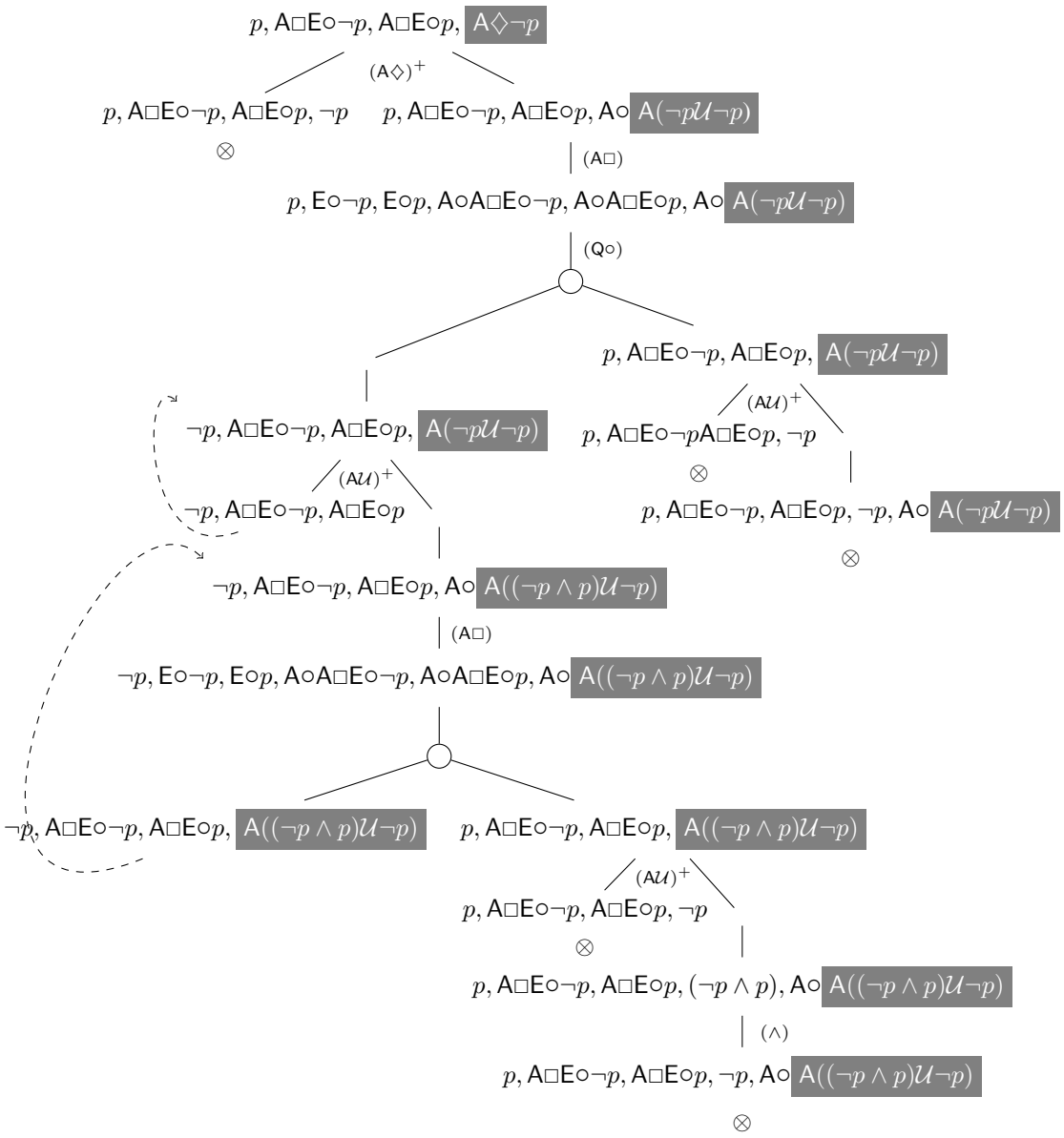

Figure 8 A closed tableau for $\{p, A\Box E\circ p, A\Box E\circ\neg p, A\Diamond\neg p\}$.